AMCA : An Active-Based Multicast Congestion Avoidance Algorithm

Moufida Maimour, Cong-Duc Pham

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

AMCA: an Active-based Multicast Congestion Avoidance Algorithm

Moufida Maimour — Cong-Duc Pham

RESO INRIA, LIP

N° 4689

Janvier 2003

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4689--FR+ENG

AMCA: an Active-based Multicast Congestion Avoidance Algorithm

Moufida Maimour* , Cong-Duc Pham†
RESO INRIA, LIP

Thème 1 — Réseaux et systèmes
Projet RESO

Rapport de recherche n° 4689 — Janvier 2003 — 29 pages

Abstract: Many works have recently addressed the issue of congestion control for multicast communications and the problem is known to be highly complex. Scalability, responsiveness, stability and fairness with TCP are some of the required properties. In this report, we present a congestion avoidance scheme for bulk data distribution called AMCA (Active-based Multicast Congestion Avoidance algorithm) that tries to meet these properties. We use the active networking technology to perform on a per-section dialogue to probe for available bandwidth along a multicast tree. The solution uses the RTT variations experienced by every branch to estimate the congestion situation in the multicast tree. The physical multicast tree is also used to appropriately aggregate the RTT variations at intermediate nodes before they reach the source. Simulations show that AMCA converges, makes use of the available bandwidth and reacts rapidly to dynamic changes while being TCP-fair. The results also show that losses are very few demonstrating the conservative property of AMCA which reacts to congestion before a loss occurs.

Key-words: active networks, reliable multicast, congestion avoidance

This work is supported in part by the french ACI Grid program and by a grant from ANVAR-EZUS Lyon.

* Moufida.Maimour@ens-lyon.fr
† Congduc.Pham@ens-lyon.fr

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 Montbonnot-St-Martin (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Un algorithme d'évitement de congestion en multicast avec les réseaux actifs

Résumé : Le contrôle de la congestion en multicast est un problème ardu. Récemment, beaucoup de recherches ont été consacrées à ce problème. Le passage à l'échelle, le temps de réponse, la stabilité et l'équité avec TCP sont parmi les propriétés requises par un algorithm de congestion. Dans ce rapport, un algorithme d'évitement de la congestion baptisé AMCA (Active-based Multicast Congestion Avoidance algorithm) qui vise la satisfaction de ces propriétés, est proposé pour la distribution des données. Nous utilisons la technologie des réseaux actifs pour permettre un dialogue par *section* pour estimer la bande passante disponible dans l'arbre de multicast. La solution utilise les variations du RTT exercées par chaque branche pour avoir une idée sur l'état de la congestion dans l'arbre du multicast. L'arbre physique de multicast est également utilisé pour agréger correctement les variations du RTT au niveau des noeuds intermédiaires avant d'atteindre la source. Des simulations ont montré que AMCA converge, utilise la bande passante disponible et réagit rapidement aux changements dynamiques tout en étant équitable avec TCP. Les résultats ont montré également que les pertes sont négligeables démontrant ainsi la propriété conservatrice de AMCA qui réagit à la congestion avant qu'une perte n'ait lieu.

Mots-clés : réseaux actifs, Multicast fiable, Contrôle de congestion

1 Introduction

Various Internet applications involve multiple parties and a one-to-many communication paradigm (multicast) is better suited to these applications than unicast. The problem of reliability has been extensively studied and many proposals exist in the literature [6, 13, 17, 18, 26, 27, 38]. However congestion appears to be the most common reason for packet loss in the Internet. The IETF [21] formulated that any reliable multicast protocol must address congestion control which is compatible with the current Internet protocols (the widely used one is TCP). In general, a congestion control consists of updating a regulation parameter (rate or window size) and enforcing that parameter so the network is prevented from congestion. In this report, we propose a congestion avoidance scheme for bulk data distribution called AMCA (standing for Active-based Multicast Congestion Avoidance algorithm). AMCA has been conceived to run in conjunction with DyRAM, an active reliable multicast protocol [20, 19]. Nevertheless, AMCA is independent from DyRAM and can be applied to any other active reliable multicast protocol as well as to unicast as an enhancement of TCP-Vegas [2]. The remainder of this report is organized as follows. A brief state of the art on the congestion control in unicast and then in multicast is presented in section 2 and 3. Afterwards a detailed description of AMCA is provided in section 4. Some simulation results are given in section 5 and section 6 concludes.

2 Unicast Congestion Control

A first congestion control algorithm known as *TCP Tahoe* has been proposed by Jacobson [15] after a series of congestion collapses experienced in 1986. Since then, several variants have been implemented. In *TCP Reno* the sender uses the loss of packets as a negative feedback to regulate its window size (used as the regulation parameter). Initially, the congestion window size is set to one and is increased by one packet per one RTT. This is the *slow start* phase which allows for probing available bandwidth by exponentially increasing the congestion window. When the window size reaches a given threshold, the source enters the *congestion avoidance* phase where the window size is increased linearly by the reciprocal of the current window size on the reception of an ACK. A loss is detected on the reception of three duplicate ACKs and the sender, as a result, reduces its congestion window by one half and enters the congestion avoidance phase. A severe congestion is considered on a timeout which causes the congestion window size to be dropped to one.

In TCP Reno, when the source does not detect congestion, it continues to increase its window size by one every one RTT. This is the reason for which TCP Reno is biased against longer delays connections. TCP Vegas [2] introduced in 1994 appears to do not suffer from delay bias and to be less oscillative than Reno while achieving better throughput and fewer losses. On one hand, TCP Vegas performs a new retransmission mechanism which reduces the time to detect lost packets from the third duplicate ACK to the first or second duplicate ACK. On the other hand, TCP Vegas is fundamentally different from TCP Reno. It uses

queuing delay as a congestion measure which can be estimated from the experienced end to end delays or alternatively from the difference between the emission and reception rates.

Attempts to model the behavior of TCP have led to equations that give the throughput of a TCP connection as a function of the loss rate, the round trip time and the packet size. These equations are known by TCP formulas. The first proposed formula is known as the TCP simple equation [22] which holds for low loss rates and, under normal operating conditions, is given by:

$$T = \frac{C\ S}{RTT\ \sqrt{p}} \tag{1}$$

where T is the throughput in bits per second and S is the data packet size in bits. C is a constant to be taken from $[0.9, 1.3]$. In the steady state, C, RTT, and S are constant. An other more accurate formula known as the complex equation has been demonstrated in [25]. Such an equation has been used to derive a simpler one to be used as the control equation for the congestion control in TFRC (TCP-Friendly Rate Control) [12]:

$$T = \frac{S}{RTT\sqrt{\frac{2p}{3}} + T_{RTO}\ (\sqrt{\frac{3p}{8}})\ p(1+32p^2)} \tag{2}$$

where T_{RTO} is the initial timeout for TCP, in seconds. This equation is used to calculate an upper bound on the throughput to be not exceeded by the source so the sending rate will not be aggressive. In a formula-based unicast congestion control, equation parameters have to be determined depending on the chosen throughput function. The loss rate p could be calculated by the receiver. The round trip time RTT could be measured at either the sender or the receiver. The initial timeout t_{RTO} can be deduced from RTT. In TFRC, the receiver calculates p and feeds it back to the source which calculates the throughput using (2). The regulation parameter adopted in TFRC is the rate instead of the window size since the TCP equation gives an upper bound for the transmission rate. One limitation of an equation-based approach is that the estimation of the loss rate remains an open issue and is difficult to do.

To improve the performances of TCP, router-assisted schemes have been proposed. For instance, RED (Random Early Detection) was proposed as a different packets drop mechanism in the routers queues. In RED, a packet is dropped with a probability that increases with the mean queue length in contrast to the traditional drop-tail where packets are dropped when the queue is full. Losses occur randomly which allows for better cooperation among the TCP sources since these latters will not react at the same time to losses. However RED increases packet loss rate even in the absence of congestion which would reduce the throughput. Moreover, the parameters of a RED mechanism are difficult to tune. TCP ECN (Explicit Congestion Notification) [28] is a variant of TCP RED where a router marks packets (with a special field in the IP header) instead of dropping them. In such a way, a receiver is able to detect its connection congestion state. The source is also able to detect a

congestion situation without losses thanks to the marks copied in the ACKs by the receiver. Moreover, in TCP ECN, a router notifies the sender allowing it to react appropriately.

More recently, active networks where routers are able to perform customized services on the packets flowing through them, have been exploited to propose more responsive congestion control algorithms. ACC (Active Congestion Control) [9] is one active scheme that makes feedback congestion control more responsive to network congestion using active networks. ACC includes necessary information in the packets such as the sender's current estimation of RTT and its window size. When a router detects a congestion, it decides what action the endpoint would take if it had detected the congestion. The router calculates the correct window size for the sender from the information provided in the packets. Afterwards the router installs a filter at the previous router on the connection's path. This filter deletes packets that the endpoint would not have sent. In meanwhile, a packet with the new calculated window size is sent to the sender which will react to the congestion accordingly. Simulations performed on ACC (compared to the traditional TCP) shows a throughput improvement up to 20% with bursty traffic and mainly in connections with high delay-bandwidth product. This is very promising since connections with high delay-bandwidth product present an unforeseen behavior. The larger the end-to-end delay in a network, the longer until the sender learns about a congestion. The higher the bandwidth of the network, the larger the amount of data the sender continues to send into a congested network before reacting to the congestion. A limitation of ACC is that it detects congestion due to losses which are only observed at the active routers. Losses experienced by other passive nodes are not taken into consideration by the active mechanism (although considered by the TCP-like end-to-end mechanism).

3 From Unicast to Multicast

When a sender has to send the same information to more than one receiver, unicast becomes inefficient and the use of multicast is more appropriate. Congestion control is then much more difficult due to the presence of multiple receivers with different end-to-end paths that could share some branches in the multicast tree. In order to be reactive to congestion situations, feedbacks from the receivers are required. A feedback could be a NACK to indicate a missed packet or an ACK packet to inform the source that a packet is correctly received. Other feedback forms with different contents can be used. In multicast, feedbacks from the receivers reflect diverse congestion conditions in various parts of the network. Feedback suppression and/or aggregation is required to be more scalable to a large number of receivers. However feedback suppression could yield to a lack of information at the sender making it very unresponsive to congestion variations. The network becomes then an open loop control system and will more difficult to be controlled. In other words, we have to deal with the feedback implosion problem while allowing the necessary information to be given to the source. A main issue of a congestion control algorithm is how, where and when such feedbacks have to be suppressed and/or aggregated.

3.1 Congestion Feedback Suppression

In contrast to unicast, in multicast there are multiple different end-to-end paths and a single loss has to be distinguished from multiple losses. A single loss may affect many receivers if their end-to-end paths to the source share the lossy link. If the source reduces its rate in response to every congestion indication it receives, its rate will be severely reduced causing the *drop-to-zero* problem [30]. Motivated by the stochastic nature of loss events particularly in a multicast Internet group, RLA (Random Listening Algorithm) [33] uses a probabilistic approach to suppress feedbacks. A loss is detected by the sender on timeout or discontinuous ACKs. The sender keeps track of the loss rates and RTTs to the different receivers. Based on these information, the sender reacts to a congestion indication with probability $1/n$ where n is the estimated number of the most congested receivers. In this way, the sender responds according to the average instead of the the maximum loss rate experienced by the receivers.

In LTRC [24], receivers estimate their average loss rate and feed it piggybacked on NACKs to the source which will change the rate accordingly. LTRC tries to react only to the most congested paths by using a threshold-based mechanism. The source would apply a big rate change only for persistent losses. LTRC reacts to only one loss indication in a time period estimated as the required time by a rate change to take effect in the system. This is due to the fact that reacting to network conditions is appropriate if these conditions are observed over a sufficiently long period of time, consisting of at least several RTTs. One issue related to this approach is how to make the tradeoff between the "drop-to-zero" problem and the responsiveness of the system. Furthermore, LTRC performs no feedback suppression before reaching the source, hence it still suffer from feedback implosion problem. Consequently, we have to address the feedback implosion problem before reaching the source by adopting a feedback suppression combined or not to an aggregation mechanism. One suppression approach consists in choosing one or more receivers to act as representatives of the whole group. These representatives would send feedbacks to the source instead of all the receivers do.

In [8], a small set of receivers are dynamically chosen as representatives by the sender based on their feedbacks. Representatives provide immediate feedback to the source allowing it to adjust the current state of the session. A probabilistic approach is used to limit feedbacks from subtrees not yet covered by the representatives. Feedback from the non-representative receivers is scheduled over a random interval. To avoid congesting the network with ACKs, representatives send ACKs with probability $1/R$ where R is the size of the representative set.

PGMCC [31] selects one receiver to act as a representative designated by the "*acker*". Every receiver estimates its capacity using the simple TCP formula (1) and feeds it back to the source piggybacked on NACKs. Based on that, the sender selects as the acker the receiver with the lowest capacity. Once the acker is chosen, a closed control loop is run between the sender and the worst receiver in the group. The acker acknowledges every received packet allowing the source to react to congestion in a TCP-like manner. NACKs suppression among the receivers is performed in a randomized manner to avoid NACK implosion.

One concern with a representative approach is to choose appropriately the representatives so they reflect the congestion situations. How to select the right size and the suitable set of representatives needs further investigation. Other schemes deal with the congestion feedback implosion problem by adopting a hierarchical structure in a similar way that reliable multicast protocols do. Feedbacks are propagated and aggregated through the tree structure. For instance, TRAM [5] and MTCP [29] are based on a logical recovery tree constructed with the receivers. An intermediate receiver node stores a received packet until it has been acked by all of its children. Feedbacks are also aggregated through these intermediate nodes.

A MTCP parent maintains two windows, a congestion window which is managed similarly to that of TCP (with slow start and congestion avoidance) and a transit window that keeps track of the amount of data that the children of a parent have not yet acked. For each ACK, a parent node transmits a congestion summary (CS) to its parent. CSs are aggregated in intermediate nodes thus avoiding the feedback implosion problem. A CS contains the minimum of its own congestion window size and those reported by its children and the maximum of its transit window size and those reported by its children. The sender is allowed to transmit the difference between the minimum congestion window and the maximum transit window.

In TRAM, the sender adjusts its rate according to receivers feedback (congestion reports). Receivers send congestion reports per one window to their parents (repair heads of the recovery tree [6]. A TRAM ACK contains the highest consecutively received packet sequence number H_r and the highest allowed sequence number H_a. Repair heads aggregate the congestion reports on the way to the sender by sending its own H_r and the minimum H_a of those reported from its children. A compromise between optimizing the speed of updating the sender's window and the potential overhead of additional ACKs is achieved by just sending ACKs where the new computed H_a^{new} satisfies $(H_r^{new} - H_a) > W_a/2$.

3.2 Fairness with TCP

As have been pointed out, feedback suppression is essential and has to be performed appropriately. Inadequate suppression of feedbacks makes the system very unresponsive to variations in the network which is a potential source of unfairness. However, fairness with other multicast flows and especially with TCP is required since most of the Internet traffic is transported by TCP. Responsiveness is more required in multicast than in unicast since the network multiplies the traffic injected by the source along the distribution tree. The larger the responsiveness, the smaller the potential oscillations and self-synchronizing behaviors are. A multicast congestion control algorithm has to be as responsive as the TCP does in order to be TCP-friendly. A reliable multicast connection is considered to be TCP-fair if it is fair with TCP flows on the worst link that it crosses. TCP is known to be window-based fair which means that each connection is given a throughput inversely proportional to its RTT. This fairness is the same as the "global fairness" introduced in [34].

To be TCP-friendly, congestion control algorithms either emulates the behavior of TCP by using a similar window-based approach or by using the TCP formula. In a formula-based approach, information about the packet loss rate and RTTs are collected. Afterwards the

TCP formula is used to compute the corresponding rate. It generally adopts a rate-based control in opposition to a window-based adopted in TCP. One of the congestion control algorithms that uses the TCP formula is TFMCC [35] which is an extension to multicast of TFRC [12] (section 2). Each receiver measures its own loss rate and its RTT to the sender and computes the corresponding rate using equation (2) as if it was connected to the sender via TCP. If a receiver sends feedback with rate less than the current emission rate then the latter is dropped to the fed back one. To avoid feedback implosion, a receiver would send its rate only if it is less than the current emission one. The receiver which has the lowest computed rate is considered as the CLR (Current Limiting Receiver). A CLR is then allowed to send feedback without any suppression, so the sender can use the CLR's feedback to increase the transmission rate. A CLR will change if another receiver sends feedback indicating that a lower transmission rate is required. The use of TCP equation is promising to be TCP fair, provided that the receivers RTTs and loss rates are known. However measuring accurately the RTTs and especially the loss rate is a real challenge. There is a trade-off to be considered between responsiveness and accuracy when trying to measure these quantities.

It is worth mentioning that measuring the loss rate experienced by every receiver can be avoided if a TCP-like window-based scheme is adopted. In MTCP [29] for instance, in addition to the source, every intermediate node manages its congestion window with a slow-start and congestion avoidance phases. The congestion window size is incremented only on receipt of ACKs from all its children in the logical tree. A TCP-Vegas like congestion control is also used in conjunction to the previous one in order to detect congestion even during slow-start. RLA [33] is an other window-based regulation scheme that achieves an acceptable degree of fairness designated by "essential fairness" in opposition to an absolute fairness very difficult to be obtained. Absolute fairness requires that the multicast session be treated as a single session and equally shares the bottleneck bandwidth with competing TCP connections on its soft bottleneck paths. According to [33], a multicast session is essentially fair to TCP if its average throughput λ in the long run is bounded by:

$$a\ \lambda^{TCP} < \lambda < b\ \lambda^{TCP}$$

where λ^{TCP} is the average throughput of the competing TCP connections on the soft bottleneck path. a and b are functions of N (number of receivers) such that $a \leq b < N$. It is worth noting that absolute fairness is a special case of essential fairness with $a = b = 1$.

3.3 Window-based versus Rate-based

As have already been seen, the regulation parameter of a congestion control algorithm is either the size of a window [29, 30, 33]) or the transmission rate [6, 24, 35]. In a window-based regulation, the source can send new packets into the Internet whenever there is space in the window. The main feature of a window-based congestion control is that it integrates naturally with error and flow control and reacts rapidly to network changes. In a rate-based approach, the source can inject new data no higher than a given rate. The main feature

of a rate-based control, generates traffic into the network smoothly and is better suited to continuous media applications. A rate-based is better suited to multicast since it is more scalable and easier to be implemented. Moreover, the rate attainable can be controlled to be independent of the RTT (hence insuring a rate-based fairness which tries to achieve equal throughput at the bottleneck link). However TCP fairness is known to be *window-based* [14] where packets tend to arrive in clusters and any different pattern (such as a rate-based regulation) will produce different loss rates through drop-tail gateways [33]. The RTT must be known to a rate-based approach to be TCP-compatible fair (window-based fairness). RTT estimation is difficult in multicast because of the problem of feedback (ACKs) implosion at the source. This problem can be overcomed using a hierarchical approach.

The proposal of [37] is an attempt to benefit from the advantages of the two approaches and combines the fast dynamics of window scheme with the flow smoothness of a rate scheme. It uses the rate as the regulation parameter. However the rate is computed by every receiver from a congestion window size updated in a TCP manner. Each receiver performs a standard TCP congestion control and updates the window size w according to the experienced losses. The transmitting rate r is then computed using $r = w/RTT$.

3.4 Receiver-driven Approaches

In congestion control algorithms, the sender (like in TCP) usually controls the rate for the whole group. Such an approach requires explicit monitoring at the source in regards to every individual receiver. On one hand this is not appropriate for most of the applications and not compatible with the IP multicast model [7]. On the other hand the larger the receivers group is, the more complex the congestion control tasks are. Furthermore because of local recovery and feedback suppression performed in many congestion control algorithms, the loss information to all the receivers is not automatically available at the source.

A step in gaining congestion control algorithms complexity is to distribute responsibilities between the sender and the receivers to be more scalable. In fact a congestion situation could be detected by the receivers instead of the source. This is mainly interesting when a drop decision has to be taken, a receiver drops itself from the multicast group if it experiences losses even when the source transmits with a specified minimum rate. The receivers could also contribute to the congestion control tasks by estimating their loss rates as in [35] (see section 3.2). In [14], an AIMD [4] window-based congestion control is run at every receiver. A receiver updates its window size and sends the necessary feedback to the source. A window at each receiver is equivalent to one window at the source with the worst representative. It is worth mentioning that multi-rate approaches (see section 3.5) are the main schemes that involve actively the receivers in the congestion control tasks.

3.5 Single-rate versus Multiple-rates

In all the presented congestion control algorithms until now, the sender controls the rate dictated by the worst receiver in response to the received feedbacks. These controllers are known as *single-rate* schemes where all the receivers get the same data rate. The main

advantage of such approach is that the source is not required to transmit multiple streams or use special data encoding. However multiple receivers in a multicast session means dealing with heterogeneity. First, in presence of large and heterogeneous groups, the problem is how to determine the slowest receiver knowing that there may be local recovery, feedback suppression and correlation in loss rates. Secondly, the worst receiver will slow down the throughput to all the receivers. Additionally as have been seen in a previous section, congestion control decisions are not always implemented by the source. For instance, a congestion control algorithm decision can be the action of dropping a receiver which must be implemented by the receiver itself.

Multi-rate schemes are based on the ability to generate the same data at different rates over multicast streams. Each receiver controls the rate at which it receives the data, usually by using multiple multicast groups. The receivers join and leave groups depending on their path congestion state so the amount of data being received is always appropriate. This is commonly known by layered multicast [32, 16]. A multi-rate scheme is more suitable to audio and video streams and reliable data transfer by using proper coding techniques. One advantage is that receivers with different needs can be served at a rate closer to their needs rather than having to much the speed of the slowest receiver. The main drawback is that its flexibility is paid in terms of coding costs and overheads relied to join and leave actions. Other limitations of such schemes consist mainly in their slow convergence time and lack of inter-protocol fairness mainly to TCP which becomes critical when the delay to join/leave a group is high.

3.6 Toward Active networking based approaches

In opposition to an end-to-end congestion control, router-assisted schemes provide additional mechanisms to multicast congestion control. Even if in the current Internet, such schemes require changes to be done in routers and are not deployed yet, are very promising. The main contribution of active routers to multicast consists in the possibility to benefit from the physical multicast tree and thus avoid the complexity inherent to building and maintaining a logical tree structure [6, 29]. More specifically, multicast protocols can benefit from special features such as feedback aggregation [17], hierarchical RTT estimation [14], cache of data [1] or the management of multiple groups in case of a layered multicast [36].

NCA (Nominee-based Congestion Avoidance) is the congestion control algorithm associated to AER (Active Error Recovery) [17]. In AER, repair servers co-located with active routers are used to cache data packets in order to insure locale recoveries when losses occur. NCA adopts a representative (nominee) strategy where the sender's congestion window is updated according to packet loss indications from this nominee. The nominee is the worst receiver in the multicast group chosen based on congestion status messages (CSM) sent periodically by the receivers. A CSM sent by a receiver to its upstream repair server contains its experienced loss rate and its RTT to the multicast source. The end-to-end loss rate p is estimated using a sliding window of a fixed size and the RTT is estimated using a hierarchical algorithm. The CSMs are aggregated at the intermediate repair servers. Every repair server determines the worst receiver according to a congestion metric ($RTT \times \sqrt{p}$) derived from the

simple TCP equation (1). A repair server maintains the identity of the worst downstream receiver, as well as the most recent loss rate and RTT estimates for that receiver and unicasts this information periodically to its closest upstream repair server. Once the source has identified the worst receiver based on the same congestion measure, it unicasts a message to this receiver soliciting per-packet acknowledgments from it. Afterwards, it reduces its rate in response to loss indications from only this receiver, and ignores those from all other receivers. The source updates its congestion window in a manner very similar to TCP New Reno [11] with some differences.

RMANP (Reliable Multicast Active Network protocol) [3] is an other congestion control that benefits from processing performed at the routers using active networks. Active routers in RMANP perform cache of data packets for potential retransmissions. Additionally, routers are able to detect congestion which makes the system reaction to congestion faster. When a congestion occurs, an active router retransmits lost data packets from its cache at a controlled rate and holds newly received ones in a queue for subsequent forwarding. When congestion decreases, an active router forwards data packets from its cache at an increasing output rate, until the cache is empty. An active router is able to detect congestion based on the status of its buffer used to store data packets which have not been absorbed by the outgoing interfaces. Performing sequence number control, the active routers and the receivers are also able to detect congestion based on their experienced losses. A congestion detected by an active router or a receiver, is indicated explicitly to the source using congestion indication (CI) messages, each of them contains a rate value computed according to the experienced losses. A CI is processed by all the active routers upstream the congestion location hence collaborating to congestion control tasks with the source in a distributed manner. CIs are filtered by active nodes based on the stamped rate avoiding multiple reaction to a single congestion instance as well as an implosion of CIs. A severe congestion is detected by the source or an active router if no FSI is observed. In such a situation, the output rate is reduced to its minimum specified value. A receiver drops itself if it considers that it can not receive data packet with the session minimum rate. Besides the congestion situations which are explicitly indicated, the source requests periodically feedback state information (FSI) from its direct active routers and those with receivers as direct descendants. A FSI contains information about the RTT of its originator to the source and its highest acked sequence number. The source will increase its output rate if the highest acked sequence number received increments the global one.

The active networks technology has also proved to be effective in a layered multicast congestion control like ALMA (Active Layered Multicast Adaptation [36]). Active networks are used in ALMA to overcome the lack of mechanisms to obtain the required information about resource availability. One ALMA feature is its fast convergence due to its capsule-based congestion indication mechanism. Active network capsules are used for bandwidth probing, and layer pruning information. ALMA is based on an economic supply and demand model. Every downstream link exports a price as a function of its characteristics and load (the average queue length). Data capsules filter themselves by comparing price and

budget and prune multicast branches that have persistent congestion. A receiver could send subscribe capsules to probe for additional bandwidth.

4 AMCA: a New Active-based Congestion Avoidance

In this report, we propose AMCA, a new active-based congestion avoidance scheme for bulk data transfer. In this study, AMCA has been integrated into the DyRAM framework [20, 19], but can be easily adopted to any other active reliable multicast protocol. AMCA is a congestion avoidance mechanism rather than a control mechanism because the sender reacts before a loss occurs. AMCA uses the RTT variation as an estimation of the number of queued packets as a congestion measure in a similar way TCP Vegas does. However it appears that TCP Vegas suffers from path changes which could give erroneous estimation of the RTTs. To avoid the problem of rerouting paths, we adopt a hop-by-hop RTT estimation and bandwidth probing. Using the active networking technology for practical implementation, a per-section dialogue is performed between adjacent nodes including the routers. A section is defined as the set of point-to-point links and traditional routers that connect two active routers or an active router to a terminal node (a receiver or the source). In AMCA, a RTT variation is measured for every section. The different RTT variations are aggregated appropriately so the source ends up by receiving the overall RTT variation experienced by the worst end-to-end path of the multicast tree.

Like RMANP [1] and NCA [17], we benefit from the physical multicast tree to perform congestion feedback aggregation and thus overcome the complexity inherent to building and maintaining a logical tree structure [6, 29]. The main difference between AMCA (rate-based) and NCA (window-based) resides in the usage of different congestion measures. Whereas AMCA uses mainly the RTT variation, NCA uses both the loss rate (p) and the RTT to determine the worst receiver (by computing the quantity $RTT\sqrt{p}$). However the loss rate is very difficult to be estimated on a short period while a sufficiently long period would make the congestion control unresponsive. Moreover, when there are no losses, all the receivers, according to their adopted measure, are considered to have the same capacity. Receivers are distinguished among them only and only if at least one receiver experiences a loss. This is also the case of RMANP which detects congestion on packet losses. In our case, the RTT variation measure in AMCA is able to predict congestion situations even in the absence of losses. Unlike RMANP, AMCA don't perform any cache of data packets in the active routers. We believe that an active router has to support multiple multicast sessions and caching means are not always available. In what follows an overview of DyRAM for which AMCA has been conceived is presented. For more details about DyRAM, the reader can refer to [20]. We show how an RTT variation measure can be used to probe for available bandwidth in a connection. Afterwards we present our mechanism adopted to estimate the different RTTs and their variations. How the feedbacks are suppressed and aggregated along the multicast tree is also dealt with. Finally, we show how the source reacts to feedbacks it receives so the multicast tree is prevented from congestion.

4.1 DyRAM : an Overview

DyRAM is a reliable multicast protocol with a recovery strategy based on a tree structure constructed on a per-packet basis with the assistance of routers. It uses a receiver-based local recovery where receivers are responsible for both the loss detection and the retransmission of repair packets when it is possible. In DyRAM, the source sends data packets to the multicast address subscribed to by all the receivers. The core functionalities of IP multicast are assumed to deliver the packets to the receivers in a best effort manner. A receiver detects a loss by sequence gaps or timeouts. Upon the detection of a loss, a receiver sends immediately a NACK toward the source and sets a timer. Since NACKs and repairs may also be lost, a receiver will re-send a similar NACK when the requested repair has not been received within the timeout interval. On the reception of a NACK packet, the source sends the repair packet to the multicast address. In order to limit the processing overheads of duplicate NACKs and to avoid the corresponding retransmissions, the source, the active routers and the receivers (remember that DyRAM uses repliers among receivers) ignore similar NACKs for a certain period of time. According to this rule, a NACK will be said "valid" if it is received for the first time or no similar NACKs are received during this discard time window. In order to perform flow and congestion control as well as memory management in addition to a more efficient replier election, ACKs are piggybacked on the NACKs. In the absence of NACKs, ACKs are also periodically piggybacked on special messages called "Congestion Reports" (CRs). DyRAM is an active reliable protocol where routers play an active role which consists in the following active services:

- the early loss detection of packet losses and the emission of the corresponding NACKs. This feature provides a low recovery delay.
- the NACK suppression of duplicate NACKs in order to limit the NACK implosion problem.
- the subcast of the repair packets only to the relevant set of receivers that have experienced a loss. This helps to limit the scope of the repairs to the affected subtree.
- the dynamic replier election which consists in choosing a link as a replier to perform a local recovery from a receiver downstream instead of caching data packets at the routers. Dynamic election provides robustness to host and link failures.

A DyRAM packet contains in its header the multicast address, the source address and the active service identifier (SVC) to be performed on this packet (see figure 1). A data packet is labelled uniquely by a sequence number and contains a dedicated field (isR) to distinguish an original transmission from a repair. A data packet contains also the current emission rate which is mainly useful to be known by the receivers in order to correctly update timeouts. A NACK or a CR packet include in addition to their originator address, their last ordered (lo) and last received (lr) data packet. Moreover, a receiver will report its RTT to the source and to its active router using the *RTT2S* and *RTT2P* fields contained in NACK and CR packets. Whereas a NACK contains the sequence number of the requested data

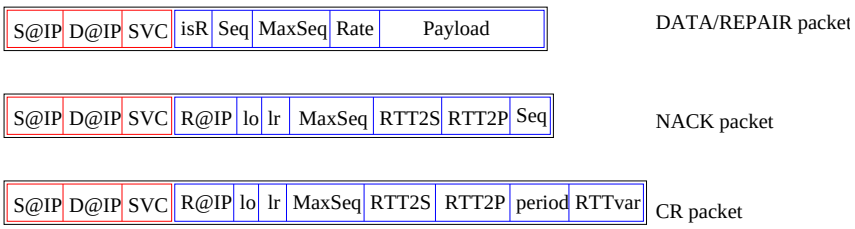


Figure 1: Packets structure.

packet, a CR contains two other fields which are the CR current period and an aggregated value of the RTT variation experienced by the subtree from which the CR is received.

For the purpose of flow control, every receiver includes in the $Maxseq$ field of a CR/NACK (figure 1), the maximum data packet sequence number that corresponds to its available buffering means. The $MaxSeq$ field in a CR or NACK packet is used by the source to control the transmission flow. An active router includes the minimum of the received $MaxSeq$ in the CR to be sent upstream to the source. The source then sends data packets with the current rate until the minimum $MaxSeq$ of all the receivers. The lo field is also used to release buffers at both the sender and the receivers. With the proposed mechanism of aggregating the CRs, the source ends-up by receiving the minimum lo of the whole group. This is an acknowledgement that all data packets up to lo have been correctly received by all the receivers. The source could then release buffers of data packets with sequence numbers up to lo. The source includes the current lo of the whole group in the $MaxSeq$ field of the data packets it sends. A receiver on the reception of a data packet would update its flow control window according to the $MaxSeq$ field of the received data packets. Moreover and since a receiver would keep data packets in order to be able to act as an eventual replier, this mechanism is helpful to release memory. A receiver could remove from its memory all the data packets with a sequence number less or equal to the $MaxSeq$ specified by the received data packet.

4.2 Probing Extra Bandwidth of a Connection

In this section, we consider the case of a connection with one queue between a sender and one receiver (figure 2). Let r_b be the bottleneck rate of the connection and r the transmission rate of the sender in bits/s. An intuitive observation is that while the transmission rate is less than the bottleneck rate, the reception rate is equal to the emission rate. The reception rate could not exceed the connection bottleneck rate r_b even if the emission rate is increased. When the transmission rate exceeds the bottleneck rate, a queue of packets will build up through the connection. Let $q(t)$ be this queue size in packets at time t. Let Δq be the positive or negative variation in the queue length during a given time period T (figure 2):

$$\Delta q = q(t+1) - q(t)$$

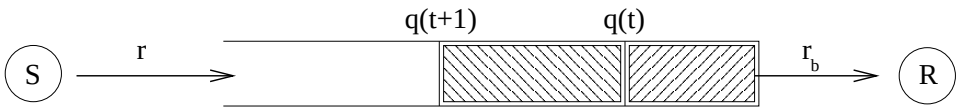


Figure 2: Queue length variation.

Noting by S the packet size, the queue builds up when $r > r_b$ during T with:

$$\Delta q = (r - r_b)\ \frac{T}{S} \tag{3}$$

To estimate Δq we can employ a mechanism based on round trip time variations since a growing up (or the opposite case) queue produces an increased (or a decreased) RTT. The RTT variation $\Delta\tau$ can be expressed in terms of Δq as follows:

$$\Delta\tau = \Delta q\ \frac{S}{r_b} \tag{4}$$

which gives:

$$r_b = \Delta q\ \frac{S}{\Delta\tau}$$

Hence,

$$r_b = r\ \frac{T}{T + \Delta\tau} \tag{5}$$

This gives an expression of the bottleneck rate and is used by our congestion avoidance algorithm to adjust the rate so the transmission rate will be the closest to the bottleneck rate. From (3) and (4) we derive the next equation which gives an expression of Δq as a function of r, T, S and $\Delta\tau$:

$$\Delta q = \frac{r\ T}{S}\ \frac{\Delta\tau}{\Delta\tau + T} \tag{6}$$

We can easily obtain accurate values for r, T and S. However for $\Delta\tau$ we have to provide an estimation mechanism. To do so in such a simple connection, the source could have an estimation of the RTT variation just by exploiting feedbacks information. For instance if we suppose that the receiver sends an ACK per packet received then the source can use the difference between the two last experienced RTTs. The T period in this case is S/r_b.

The problem when using such a method to probe available bandwidth appears when we have a unicast (or multicast) session with multiple links. The path between the sender and

the receiver can change and this would give erroneous results. [23] pointed out that TCP Vegas suffers from similar situations since rerouting a path may change the propagation delay of the connection which could result in a substantial increase/decrease in throughput. To overcome this problem, we propose to use a per-section bandwidth probing approach. A section is defined as the set of point-to-point links and traditional routers that connect two active routers or an active router to a terminal node (a receiver or the source). The main concern is to be able to estimate the RTT variations in a per-section fashion and then how these RTT variations are aggregated before giving them to the source. To do so, we have to establish communications between every node and its neighbor in a connection.

4.3 RTT Estimation

A straightforward solution to estimate the RTTs between each receiver and the source consists in sending ping messages periodically from the receivers to the source. This naive solution would however overload the source in the presence of a large number of receivers. Instead of estimating directly the RTTs between the source and each receiver, we begin by estimating the RTTs between every node and its parent in the multicast tree. This mechanism meets also the per-section RTT measurements required by our hop-by-hop bandwidth probing (section 4.2). To illustrate our method we show in figure 3, one linear path of the multicast tree that connects the sender (labeled by 0) to one of the receivers with label n. Intermediate nodes labeled from 1 to $n-1$ are the routers. The generalization of our method to the other receivers and intermediate nodes is straightforward.

Special messages called heartbeat messages (HB and HB_RESP) are periodically exchanged between each node i and its parent $(i-1)$ in the same way the ping/pong messages do. The first HB message is sent by the receiver. The reception of a HB by a router triggers the emission of a HB message to its parent in addition to responding with a HB_RESP message. Let $\theta_{i,j}$ be the computed RTT between the ith and the jth node (we have $\forall i,j : \theta_{i,j} = \theta_{j,i}$). Consequently, $\theta_{i-1,i}$ is the RTT between the node i and its parent and $\theta_{0,i}$ is the RTT between a node i and the source.

A node i, in order to compute its RTT to the source, first computes its RTT to its parent node by sending a HB message timestamped with the emission time. The parent node, on the reception of the HB message, responds with a HB_RESP that contains the its RTT to the source that has already been computed. Node i, on receipt of the HB_RESP message is then able to compute its RTT to the source as the sum of the RTT of its parent to the source and its own RTT to its parent as follows:

$$\forall i > 0 : \theta_{0,i} = \theta_{0,i-1} + \theta_{i-1,i} \tag{7}$$

Initially (right side of figure 3), the source responds with a special message HB_RESP_S which contains its own RTT to itself (of course we have $\theta_{0,0} = 0$) in the data packets it sends. Node number 1 will update its $\theta_{0,1}$ using (7) and forwards it downstream with the newly computed RTT. Node number 2 will in turn update its $\theta_{0,2}$ using the same equation and

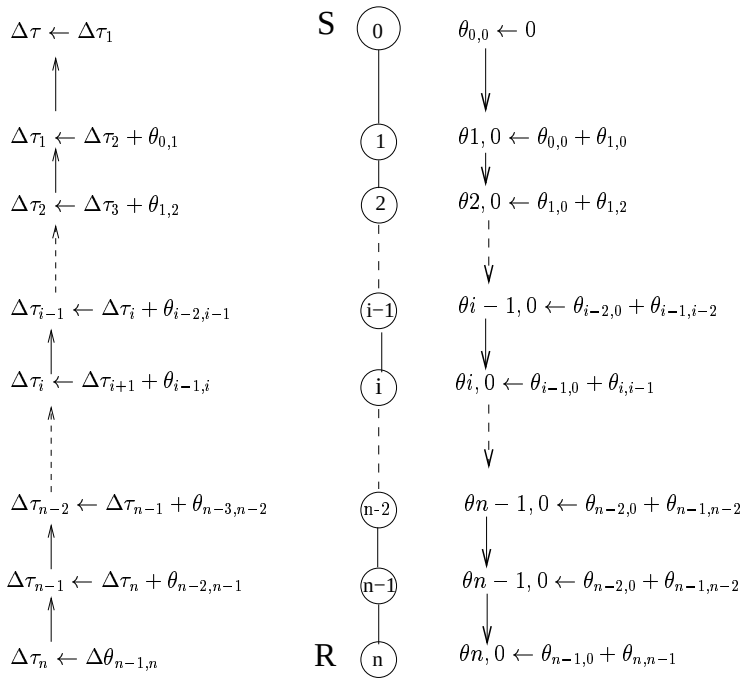


Figure 3: RTT estimation and CRs aggregation

forwards it downstream and so on (figure 3). Afterwards the use of the HB and HB_RESP messages will be sufficient to estimate the different RTTs.

4.4 Congestion Feedback

In AMCA, every receiver sends a CR to the source every N packets received so the source can learn about the congestion situation in the multicast tree. We chose to send a CR on the reception of a given number of packets rather than sending a CR every a constant period. The congestion situation is then probed proportionally to the emission rate. The resulting period is $T = NS/r$ which represents the expected time to receive N data packets by a receiver where r is the emission rate of the source. In the case where no data packet is received, a NACK could be sent instead of a CR to indicate that a packet is missed by a receiver. With $T = NS/r$, from (6) the queue size variation can be given by:

$$\Delta q = N \frac{\Delta\tau}{\Delta\tau + T} \tag{8}$$

We introduce Δq_p defined as the queue size variation per packet:

$$\Delta q_p = \frac{\Delta q}{N} = \frac{\Delta \tau}{\Delta \tau + T} \tag{9}$$

The periodicity of sending the CRs varies because it depends on the current reception rate. To make the source aware of the current period, a receiver reports this latter in the *period* field of a CR (figure 1).

In what follows, we note by $\Delta\theta_{i,j}$ the experienced RTT variation between the ith and the jth node. For the ith link that connects the $(i-1)$th and the ith node, the RTT variation is $\Delta\theta_{i-1,i}$. Additionally we note by $\Delta\tau_i$ the RTT variation of the $(i-1)$th intermediate node and the receiver. The overall experienced RTT variation between the source and the receiver is $\Delta\tau = \Delta\tau_1$. Initially, a receiver (left side of figure 3) computes the RTT variation $\Delta\theta_{n-1,n}$ of its upstream link (number n) using the last two RTT measures to its parent with the mechanism described in section 4.3. Afterwards the receiver includes the computed RTT variation in the CR to be sent to the source. The receiver's parent (i.e. node $n-1$) adds the RTT variation reported by the receiver and its own RTT variation to its upstream node to obtain the RTT variation of its parent ($(n-2)$th node) to the receiver:

$$\Delta\tau_{n-1} = \Delta\tau_n + \Delta\theta_{n-2,n-1}$$

Node $(n-1)$ will then report $\Delta\tau_{n-1}$ in the *RTTvar* field of the CR. In this way, a parent node i, on receipt of a CR from node $(i+1)$ adds its RTT variation to node $(i-1)$ to the received one before forwarding the CR with the newly computed RTT variation:

$$\Delta\tau_i = \Delta\tau_{i+1} + \Delta\theta_{i-1,i}$$

The sender will end up by receiving the RTT variation experienced on the whole linear connection $\Delta\tau$. This latter can be used in equation (5) or (9) to have respectively a lower bound for the bottleneck rate r_b or the estimated queue length variation per packet Δq_p during the previous period.

4.5 Feedback Suppression/Aggregation

To achieve scalability, a mechanism of aggregation/suppression is performed. We use the physical multicast tree to hierarchically aggregating feedbacks at the intermediate nodes (routers). A suppression mechanism is performed on NACKs thus allowing just one NACK per loss to be sent upstream. Since we do not suppose that all the routers are active, more than one NACK for the same loss could be received by the source. To avoid reacting to duplicate NACKs, the source reacts to the first one and ignores the subsequent ones for a given period of time which depends on the multicast tree structure.

The CRs are aggregated at the intermediate nodes. An active router aggregates the received CRs from the downstream links in one CR to be forwarded upstream. The aggregated CR contains the sequence number of the minimum last ordered and the maximum last received data packets among those reported by the CRs received from downstream. For

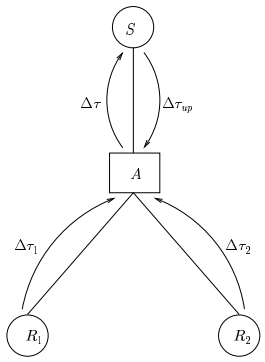


Figure 4: CRs aggregation in a multicast tree

the aggregation of the RTT variations reported by the CRs from downstream in the case of a multi-path connection, we illustrate our solution with the simple two-level tree depicted in figure 4. We consider two receivers R_1 and R_2 connected to the source S via one active router A. Receivers R_1 and R_2 send CRs to the active router with their respective RTT variations $\Delta\tau_1$ and $\Delta\tau_2$. Once these children CRs are received, the active router A sends its CR to the source with $\Delta\tau$ computed using:

$$\Delta\tau = \Delta\tau_{up} + Max(\Delta\tau_1, \Delta\tau_2)$$

where $\Delta\tau_{up}$ is the RTT variation of the source link (A, S). This latter is extracted from the HB_RESP message sent by the source. Using this method in aggregating the RTT variation, the source ends up by receiving the RTT variation experienced by the worst end-to-end path of the multicast tree.

4.6 Rate Regulation

In AMCA, the regulation parameter is the rate. A minimum and a maximum rates r_{min} and r_{max} are set by the application. Any receiver that could not support the minimum transmission rate has to leave the multicast session. Initially, the source starts to send data packets with a rate equal to the minimum rate r_{min}. Then it tries to increase its rate if no congestion indication is received without exceeding the maximum rate r_{max}. The source uses the information fed back in the CRs and NACKs to update its rate. The RTT variation field of every CR is used by the source to compute the queue size variation per packet Δq_p during the previous period. The goal is to maintain Δq_p as small as possible. During a phase similar to the slow start phase of TCP, the source would increase its rate by S/RTT_{max} (bits per second) every time it receives a CR that gives a queue size variation $\Delta q_p < \epsilon$, where ϵ is a positive number to be chosen in $[0, 1]$. Increasing the rate by S/RTT_{max} where RTT_{max} is the maximum experienced RTT among the receivers, is equivalent to adding 1 to the congestion window for the largest end-to-end connection of the multicast tree. This

behavior makes our congestion avoidance algorithm fair with TCP from the beginning of the multicast session.

However, the network could be already congested from the beginning of the session. To avoid aggravating the situation, we need an other congestion indicator to be sure that there is no congestion in the network. This can be done by examining the successive values of the last ordered field of the CRs. This gives a measure of the difference between the emission and the reception rate. This difference is proportional to the number of data packets which are not acked yet which can be given by:

$$\Delta q_a = \max(0, n_s - n_a) \tag{10}$$

where n_s and n_a are respectively the number of sent and acked packets during the previous period. Since a CR is sent to the source every N packets received during a period T, we have:

$$n_s = \frac{T}{S}\, r = N$$

The number of acked packets is computed using:

$$n_a = lo_{i+1} - lo_i$$

where lo_{i+1} and lo_i are respectively the values of the lo field of the newly received and previously received CR. Finally:

$$\Delta q_a = \max(0, N - (lo_{i+1} - lo_i)) \tag{11}$$

The source, on the reception of a CR, extracts the current period (T) and the RTT variation ($\Delta\tau$). Afterwards, in addition to the queue size variation per packet Δq_p, the number of data packets not acked yet Δq_a is also computed using (11). The aim is to maintain Δq_a in $[\alpha, \beta]$, where α and β are similar to the two parameters of TCP-Vegas. In the slow start phase, the source continues increasing its rate every time it receives a CR that indicates $\Delta q_p < \epsilon$ and $\Delta q_a < \beta$. The source enters the congestion avoidance phase when it receives a NACK or a CR with $\Delta q_p > \epsilon$ or $\Delta q_a > \beta$. On the receipt of a NACK, the source reduces the rate by half. Subsequent NACKs are ignored during a period estimated by the difference between the current RTT to the source of the farthest and the closest receiver in the multicast tree ($RTT_{max} - RTT_{min}$). To avoid decreasing dramatically the rate because of isolated losses, we check the lo and lr fields of the NACKs and reduce the rate only if $lr - lo \geq 3$, otherwise we just retransmit the corresponding repair.

During the congestion avoidance phase, on the reception of a CR with a RTT variation $\Delta\tau$ and a period T, the source updates its rate depending on the current values of both Δq_p and Δq_a. When these two measures do not indicate congestion ($\Delta q_p \in [0, \epsilon]$ and $\Delta q_a < \beta$), the rate is multiplied by γ_{inc} chosen slightly greater than 1. In the other cases, the rate is multiplied by $\gamma = T/(T + \Delta\tau)$ which is derived from (5) and allows the rate to converge to

the bottleneck rate. When the emission rate is greater than the bottleneck rate, a queue will build up in the bottleneck link. This will be translated by a positive RTT variation $\Delta\tau$ and thus $\gamma < 1$. Multiplying by γ will decrease the transmission rate and this is the appropriate action from the source. In the reverse situation where the emission rate is less than the available bandwidth, $\Delta\tau$ is negative and multiplying by $\gamma > 1$ will increase the transmission rate. In addition to the proposed mechanisms, a severe congestion is detected if no CR, HB or NACK is received during a relatively long period which is set to twice the current period ($2T$). In this case the rate is dropped to its minimum value r_{min}.

Since DyRAM implements a local recovery mechanism, we have cared about the local recovery problem. Many reliable protocols perform local recovery which leads to the coexistence of multiple data sources. This is why not only the original source has to be congestion controlled, but retransmitting entities (receivers, routers or servers) should also be controlled. Additionally, a feedback suppression is implicitly or explicitly performed which would make the source unresponsive to congestion variations. Suppose that behind a congested link, one or more receivers have sent NACKs that will be received by a replier instead of the source. The replier will retransmit the missed packets and the source is never aware about the congestion and would continue to send with the same rate and even with a higher rate. As a result the congestion situation is aggravated. This is the so called *local recovery problem* [24]. In our scheme, we propose to make the source aware about any NACK that has been forwarded downstream to an elected replier. An active router when aggregating the CRs, includes NACK information mainly in the lo and lr fields of the aggregated CRs. In this way, the source is always aware about losses that occur in the multicast tree even in the presence of local recoveries. The retransmission rate from the repliers is also controlled to do not aggravate a congestion situation until the source adapts its rate. When an elected replier has to retransmit more than one packet, then it sends repairs with a rate equal to the current rate of the source divided by $S(lr - lo)$ which is proportional to the congestion degree. Finally, it is worth mentioning that the DyRAM loss detection service at the routers could be very helpful for the congestion control. A NACK is sent to the source as soon as a loss is detected and the source would reduce its rate rapidly which could enhance the responsiveness of the proposed congestion avoidance algorithm.

5 Simulation Results

A simulation model of AMCA with DyRAM has been implemented using ns-2.1b8 (network simulator [10]). For all the simulations the data packet size is set to 1024 bytes and the NACK/CR packet size is set to 32 bytes. The minimum and maximum rates are set respectively to 1 and 10000 packets per second. While ϵ is set to 0.2, α and β are respectively set to 1 and 3. For the rate regulation, γ_{inc} is set to 1.05. A receiver sends a CR every $N = 32$ data packets received unless said otherwise. For all the simulations, we used the simple topology of figure 5 with four receivers separated by two routers from the source. The bottleneck link L has a bandwidth of $0.9Mbps$. S is the DyRAM source that multicasts data to the receivers $R1$ and $R2$. $S1$ and $S2$ are used as sources for TCP (Vegas) flows

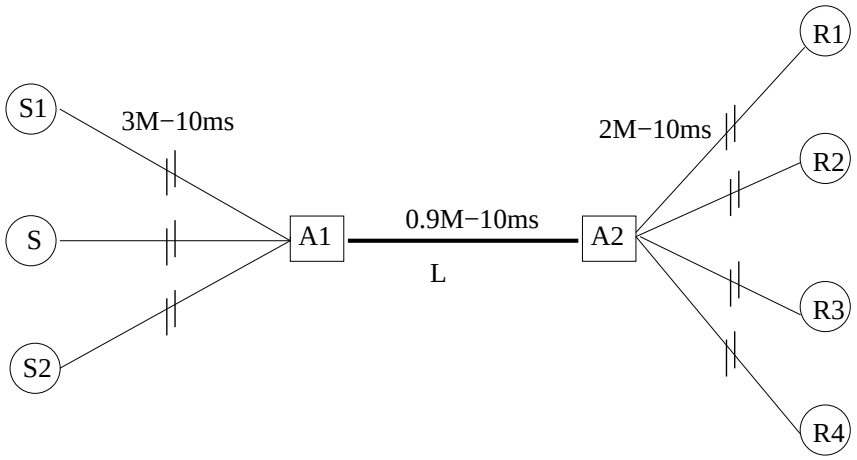


Figure 5: The used topology

to study the fairness of DyRAM with TCP. $R3$ and $R4$ are the TCP sinks of $S1$ and $S2$ respectively.

5.1 Convergence of the Algorithm

A first set of experiments is conducted with just one multicast flow. The source $S1$ multicasts data using DyRAM to the receivers of figure 5. The purpose of this first set of experiments is to show how our congestion algorithm probes and makes use of the available bandwidth. Figure 6 shows the achieved throughput by the receivers for different values for N. We can see that the algorithm converges to the bottleneck available bandwidth of $900Kbps$ and then makes use of the available bandwidth during all the simulation time. It is worth showing that there is a tradeoff between the responsiveness (quick reaction) and the stability (in terms of oscillations) depending on the CRs periodicity. We can see that the smaller N is, the more responsive the algorithm is. However the oscillations are smoother when we increase N.

5.2 Fairness with TCP

Since most of the Internet traffic is transported by TCP, the fairness to TCP flows is an important issue. To evaluate the fairness of our scheme, experiments have been conducted on the topology depicted in figure 5. One experiment is performed with one DyRAM flow competing with two TCP flows. The DyRAM source is S and the 2 TCP flows are sent by $S1$ and $S2$ to the receivers $R3$ and $R4$ respectively. The simulation is run for 200 seconds. All of the three flows are started at the same time. Figure 7 shows the achieved throughput

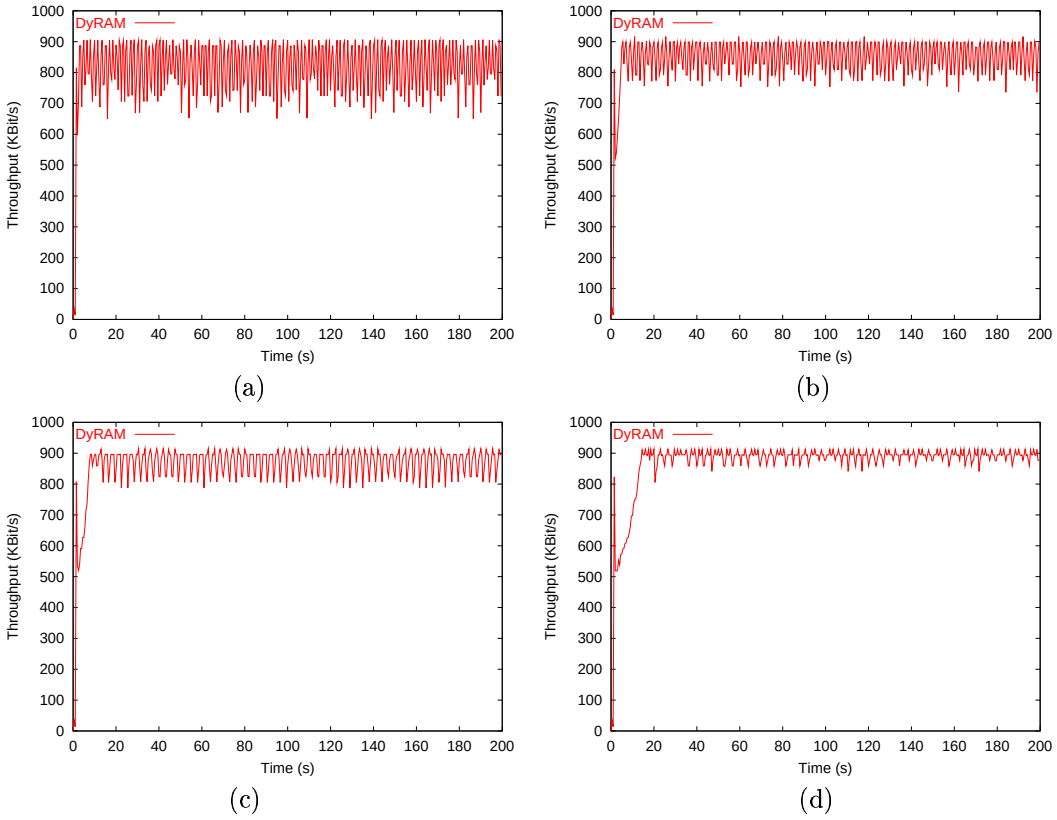


Figure 6: The tradeoff between responsiveness and stability (a) 16, (b) 32, (c) 64, (d) 128

by the different flows. We can see that the bottleneck link of $0.9M$ is almost equally shared among the three flows.

To show the dynamic behavior of our congestion algorithm, we have run a simulation with one DyRAM multicast flow in the background and one TCP flow started at 20s and stopped at 100s. Figure 8(a) shows the achieved throughput by DyRAM and TCP. We can see that DyRAM makes use of all the available bandwidth while the TCP flow is not started yet. In the presence of the two flows, the available bandwidth is shared equally among them. When the TCP flow is stopped, DyRAM is able again to make use of the available bandwidth. We note that the required time by DyRAM to react to both the existence and the absence of the TCP flow, is very small. This shows the responsiveness of our proposed algorithm. Figure 8(b) shows the sequence numbers of data packets received by the DyRAM receivers. We can observe that DyRAM dynamically adjusts its rate and the slope angle

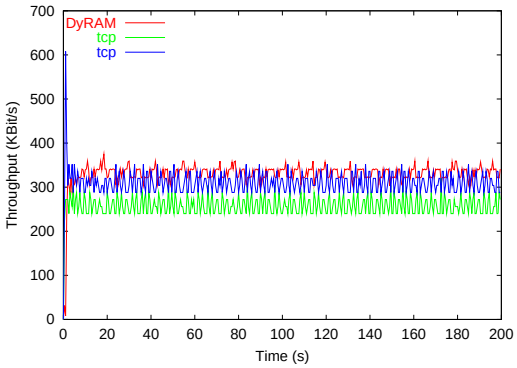
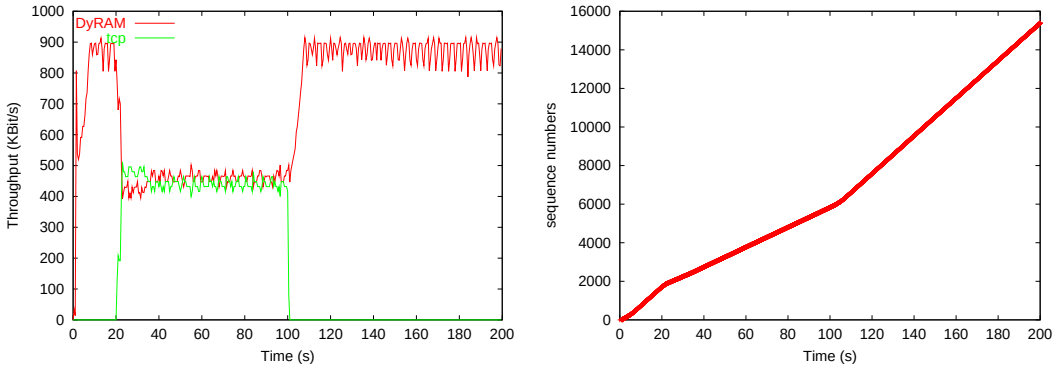


Figure 7: Fairness with TCP

of the sequence number line is divided by 2 in the presence of the TCP flow. An other interesting observation is that there is no packet loss during all the simulation time.

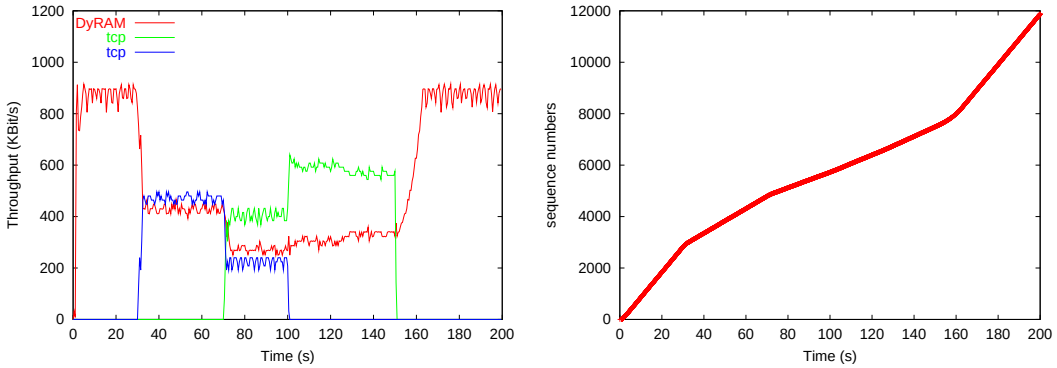


(a) (b)

Figure 8: Dynamic behavior of DyRAM with one TCP flow

Figure 9 shows the results of a simulation run with two TCP flows and one DyRAM multicast flow. DyRAM is run from 0 to 200s and the two TCP flows are run respectively from 30 to 100s and 70 to 150s. We can note the same observations as for the previous experiment. DyRAM equally shares the available bandwidth with two TCP flows and always react quickly to the network dynamics. DyRAM is always fair and never aggressive with TCP.

An other simulation has been performed with the TCP flow in the background and the DyRAM flow run from 20 to approximately 118s (the required time to send 5000 data packets). Figure 10 shows the achieved throughput by the two flows. It is clear that once



(a) (b)

Figure 9: Dynamic behavior of DyRAM with two TCP flows

more DyRAM is proved to be TCP-friendly and shares equally resources with any TCP flow.

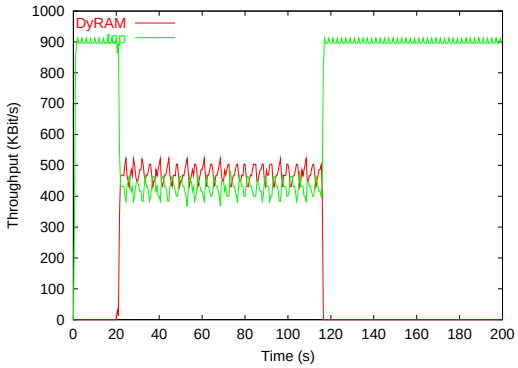


Figure 10: Fairness with one TCP flow in the background

5.3 Multiple Bottleneck Links

To see the behavior of our congestion avoidance algorithm with TCP flows in the presence of multiple bottleneck links, we have performed simulations with the same topology of figure 5 with some lower capacity links. The simulations are performed with 2 TCP flows and one DyRAM flow. The TCP flows are from S1 and S2 to R3 and R4 respectively. S is the DyRAM source that multicasts to the receivers R1 and R2. We set the link bandwidth of receivers R2 (one of the DyRAM flow receivers) and R4 to $0.2Mbps$ instead of $2Mbps$.

Figure 11(a) shows that DyRAM achieves a throughput that corresponds to the minimum available on the whole multicast tree ($0.2Mbps$). The TCP flow to R4 uses the remaining bandwidth of the link L. Figure 11(b) shows the achieved throughput by the different flows where the bandwidth of the TCP receivers (R3 and R4) links is set to $0.2Mbps$. We can see that the 2 TCP flows use the available $0.2Mbps$ bandwidth for each of them and that DyRAM makes use of the remaining bandwidth on the bottleneck link.

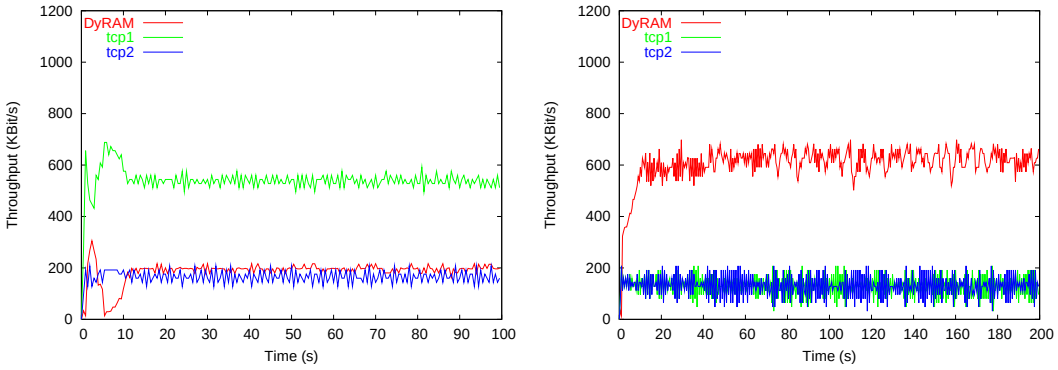


Figure 11: Multiple bottleneck links : (a) R1,R4(2M),R2,R3(0.2M), (b) R1,R2(2M), R3,R4(0.2M)

6 Conclusion

In this paper, we have presented AMCA an active-based congestion avoidance algorithm for multicast communications. The proposed scheme is validated with an active reliable multicast protocol (DyRAM). Nevertheless AMCA can be used by any other active reliable multicast protocol and particularly in the case of unicast as an enhancement of TCP Vegas. Our congestion algorithm relies on the active networking technology to perform a per-section dialogue to probe for available bandwidth along a multicast tree. This is done by using the RTT variations experienced by every section in the multicast tree. The RTT variations are then appropriately aggregated by intermediate nodes before they are given to the source.

A set of simulations have been conducted using ns-2.1b8 network simulator to study the behavior of our congestion avoidance algorithm. It has been shown that AMCA converges and makes use of the available bandwidth and reacts rapidly to the dynamic changes experienced by the multicast tree. In addition, simulations have shown that losses are very few showing the conservative property of AMCA which reacts to congestion before a loss occurs. Moreover our algorithm is proved to be fair with TCP.

AMCA has to be experimented on larger topologies that reflect the actual Internet and the congestion parameters that we used have to be studied in more detail to determine their

optimal values. AMCA is a single-rate approach and the emission rate is adapted according to the worst receiver in the multicast group. Currently an adaptation of AMCA to support heterogeneous receivers is under study and results will be available soon.

References

[1] Arturo Azcorra, María Calderón, Marifeli Sedano, and José I. Moreno. Multicast congestion control for active network services. *European Transactions in Telecommunications*, 10(3), Mau/June 1999.

[2] L. Brakmo and L. Peterson. End to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communications*, 13(8), October 1995.

[3] M. Calderón, M. Sedano, A. Azcorra, and C. Alonso. Active networks support for multicast applications. *IEEE Networks*, May/June 1998.

[4] D. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestions avoidance in computer networks. *Computer Networks and ISDN Systems*, 17:1–14, 1989.

[5] D. M. Chiu, M. Kadansky, and J. Provino. A congestion control algorithm for tree-based reliable multicast protocols. Technical Report TR-2001-97, Sun Microsystems Labs, 2001.

[6] Dah Ming Chiu, Stephen Hurst, Miriam Kadansky, and Joseph Wesley. Tram: A tree-based reliable multicast protocol. Technical Report TR-98-66, SUN, July 1998.

[7] S. E. Deering and D. R. Cheriton. Multicast routing in datagrams internetworks and extended lans. In *ACM Transactions on Computer Systems*, May 1990.

[8] D. DeLucia and K. Obraczka. Multicast feedback suppression using representatives. In *INFOCOM*, pages 463–470, 1997.

[9] T. Faber. Experience with active congestion control. In *DARPA ACtive Networks Conference and Exposition, San Francisco, CA*, May 29-30 2002.

[10] K. Fall, K. varadhan, and S. floyd. Ns notes and documentation ucb/lbnl/vint. software and documentation available at http://www.isi.edi/nsnam/ns, July 1999.

[11] S. Floyd and T. Henderson. The newreno modification to tcp's fast recovery algorithm, rfc 2582, 1999.

[12] Sally Floyd, Mark Handley, Jitendra Padhye, and Jorg Widmer. Equation-based congestion control for unicast applications. In *SIGCOMM 2000*, pages 43–56, Stockholm, Sweden, August 2000.

[13] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6), 1997.

[14] S. Jamaloddin Golestani and Krishan K. Sabnani. Fundamental observations on multicast congestion control in the internet. In *INFOCOM*, pages 990–1000, 1999.

[15] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM '88*, 1988.

[16] V. Jacobson, S. McCanne, and M. Vetterl. Receiver-driven layered multicast. In *ACM SIGCOMM'96, Stanford, CA*, pages 117–130, August 1996.

[17] S. Kasera and S. Bhattacharya. Scalabe fair reliable multicast using active services. *IEEE Network Magazine's Special Issue on Multicast*, 2000.

[18] L. Lehman, S. Garland, and D. Tennehouse. Active reliable multicast. In *Proc. of the IEEE INFOCOM, San Francisco, CA*, March 1998.

[19] M. Maimour and C. Pham. Dynamic replier active reliable multicast (dyram). In *Proc. of the 7th IEEE Symp. on Comp. and Comm. (ISCC 2002)*, July 2002.

[20] M. Maimour and C. Pham. Dyram : A reliable multicast protocol. Technical Report RR-4635, INRIA, 2002.

[21] A. Mankin, A. Romanov, S. Bradner, and V. Paxson. Ietf criteria for evaluating reliable multicast transport and application protocols, rfc 2357, June 1998.

[22] M. Mathis, J, Semke, J. Madhavi, and T. Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. *Computer Communication Review*, 27(3):62–82, 199.

[23] Jeonghoon Mo, Richard J. La, Venkat Anantharam, and Jean C. Walrand". Analysis and comparison of TCP reno and vegas. In *INFOCOM*, pages 1556–1563, 1999.

[24] T. Montgomery. A loss tolerant rate controller for reliable multicast. Technical Report NASA-IVV-97-011, NASA/WVU, August 1997.

[25] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling tcp throughput: a simple model and its empirical validation. In *SIGCOMM*, 1998.

[26] Christos Papadopoulos, Guru M. Parulkar, and George Varghese. An error control scheme for large-scale multicast applications. In *Proc. of the IEEE INFOCOM*, March 1998.

[27] S. Paul and K. Sabnani. Reliable multicast transport protocol (RMTP). *IEEE JSAC, Spec. Issue on Network Support for Multipoint Communications*, 15(3), April 1997.

[28] K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification (ecn) to ipv6 and to tcp, October 1998.

[29] Injong Rhee, Nallathambi Ballaguru, and George N. Rouskas. Mtcp : Scalable tcp-like congestion control for reliable multicast. Technical Report TR-98-01, 1998.

[30] L. Rizzo. pgmcc: a tcp-friendly single-rate multicast. In *SIGCOMM*, pages 17–28, 2000.

[31] T. Speakman et al. Pgm reliable transport protocol specification. internet draft, 1998.

[32] L. Vicisano, L. Rizzo, and J. Crowcroft. Tcp-like congestion control for layered multicast data transfer. In *Conference on Computer Communications (IEEE Infocom'98), San Fransisco, USA*, pages 1–8, 1998.

[33] Huayan Amy Wang and Mischa Schwartz. Achieving bounded fairness for multicast and TCP traffic in the internet. In *SIGCOMM*, pages 81–92, 1998.

[34] B. Whetten and J. Conlan. A rate based congestion control scheme for reliable multicast. Technical report, Reliable Multicast Research Groupworking document,, December 1998.

[35] Jörg Widmer and Mark Handley. Extending equation-based congestion control to multicast applications. In *ACM SIGCOMM (San Diego,CA)*, August 2001.

[36] Lidia Yamamoto and Guy Leduc. An active layered multicast adaptation protocol. In *IWAN*, pages 179–194, 2000.

[37] Y. R. Yang and S. S. Lam. Internet multicast congestion control: A survey. In *ICT 2000, Acapulco, Mexico*, May 2000.

[38] Rajendra Yavatkar, James Griffoen, and Madhu Sudan. A reliable dissemination protocol for interactive collaborative applications. In *ACM Multimedia*, 1995.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399